Label Embedded Treemapping: A Label Overlap Prevention Technique for Zoomable Treemaps and a User Interaction Technique

KwangHyuk Kim[1,2] and JungHyun Han[2]

[1] Samsung Electronics Co Ltd, Korea
[2] Dept. of Computer and Radio Communications Engineering, Korea University, Seoul, Korea
kwanghyuk.a.kim@gmail.com

Abstract. Data navigation of a treemap—a widely used tool for visualizing tree data—becomes more difficult as the amount of data increases. To solve this problem, treemap techniques using zoomable user interface (ZUI) methods—the most typical of which is the zoomable treemap (ZTM)—have been proposed. However, ZTMs can incur face text overlapping issues between examined nodes. In order to increase ZTM readability, we propose a label embedded tree map technique that prevents label overlapping and a direct node selection method for the highlighting of focused parent nodes. The proposed tree map technique resolves the ZTM label conflict and the direct node selection method can efficiently improve data navigation.

Keywords: Visualization, Treemap, Zoomable, ZUI, Label, Overlapping.

1 Introduction

Two visualization techniques are commonly used to present tree data: node linking [1] and treemapping [2]. In node linking, hierarchical structures of data are represented as sets of nodes interconnected by lines, while treemapping is a space-partitioning method in which the screen space is divided into adjacent rectangles that represent data [3,4]. Whereas node linking does not fully use the screen space, treemapping can maximize space efficiency because the method displays nodes as a geological map-like structure. However, treemaps begin to incur perception problems [5] with increasing data size (Fig. 1), which makes navigation through larger treemaps more difficult. Various techniques have been proposed to solve the navigation problem, including the treemap partitioning algorithm, a technique that has produced exciting results and which has been the most popular area of research. However, the most typical partitioning method, known as slice-and-dice layout [1], can essentially only divide the screen vertically or horizontally, which results in rectangles (tiles) with aspect ratios that differ significantly from one, i.e., from a square shape. A proposed squarified layout technique [6] can produce tiles with more suitable aspect ratios but does not consider the ordering of the tiles; to do this, strip treemapping using a partitioning algorithm [7,8] has been suggested, but this method does not produce improved data navigation as the data size increases.

S. Yamamoto (Ed.): HIMI 2014, Part I, LNCS 8521, pp. 44–53, 2014.

Other approaches for improving data navigation involve the use of visualization effects. Cushion treemapping [9] uses a shading effect to improve the hierarchical perception of tree data and a color lens [10] to assign a unique color to the focused area, but again, this technique does not appear to provide a suitable navigation solution for large data sets.

Approaches utilizing the distortion effect have also been proposed. K. Shi et al. [11] suggested a distortion treemap in which the focused node is enlarged while surrounding nodes are reduced, and the balloon focus technique [12] extends the distortion effect to multiple focus nodes. Fish eye treemapping [13] involves the use of a lens effect in order to enlarge the focused area. Nevertheless, as these methods do not use the full screen area for the focused node, none of them can produce the most space efficient solutions.

Zoomable user interfaces (ZUIs) [14] can provide information progressively depending on the zoom level (tree level). A ZUI integrated treemap is a photo mesa [15] that extends specific regions of interest but cannot represent the hierarchical relationships between parents and children. The University of Maryland (UMD) treemap [16] represents hierarchical relationships by allocating separate spaces to label parent (or container) nodes and also extends the focal node to the full screen. However, because the layout changes when the zoom interaction ends, the UMD treemap does not produce smooth transition animation during zooming, making it necessary to use a separate layout algorithm to allocate the parent's label information. Although the zoomable treemap (ZTM) [17] overcomes the layout and transition animation problems of the UMD treemap, many of its labels will overlap owing to conflicts between leaf and upper side parent nodes (Fig. 2). In this paper, we present a novel treemapping technique that avoids overlapping between nodes as well as a user interaction technique that assures the direct node selection and smooth continuous treemap transition animation.

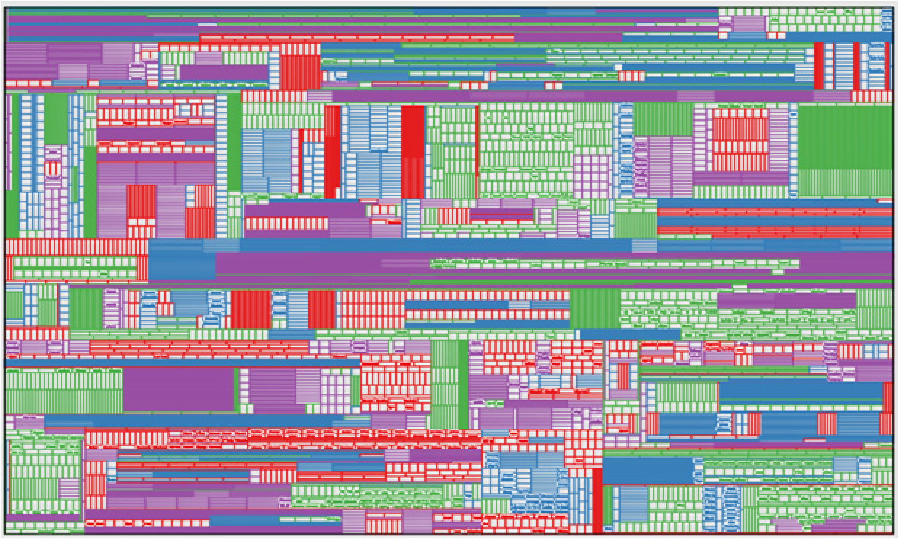

Fig. 1. Example of the treemap perception problem

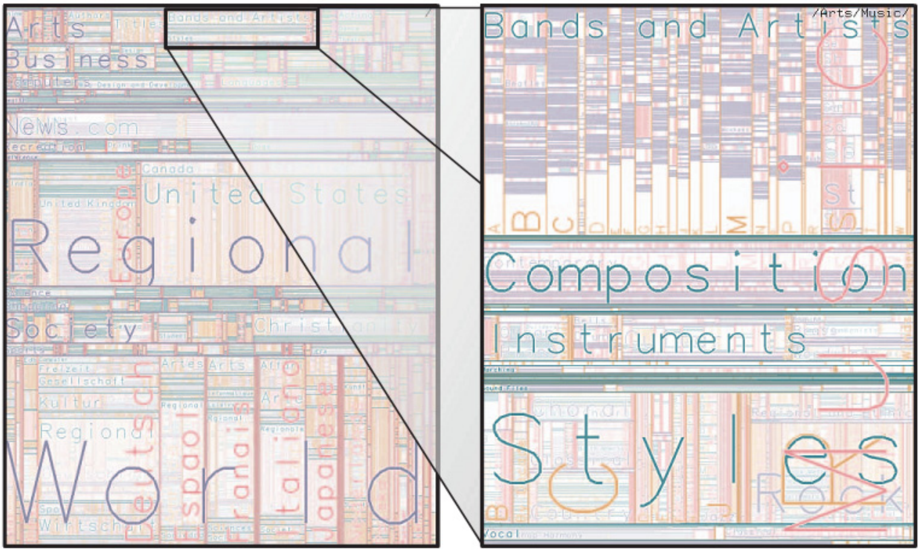

Fig. 2. Label overlapping problem using ZTM

2 Label Embedded Treemap

An initial concept and a prototype implementation of the label embedded technique were presented at a domestic conference [18]; a more accurate explanation of the proposed visualization process is shown in Fig. 3, which describes the parsing, label embedding, weight calculating, partitioning, and rendering sequences. Following the parsing stage, tree data are used to construct the tree structure, which is translated from a general structure into a label embedded tree in the label embedding stage. After a tree structure is constructed, weight value for the newly inserted node is calculated. The position of each rectangle in the treemap is set in the partitioning stage, and the nodes are rendered based on their hierarchical relationship by individually visiting the rectangles in the rendering stage. To construct the label embedded treemap, the treemap partitioning and rendering algorithm is used without modification. Through this mechanism, a label embedded tree can be automatically translated without modification into a label embedded treemap by using the partitioning mechanism of the treemap.

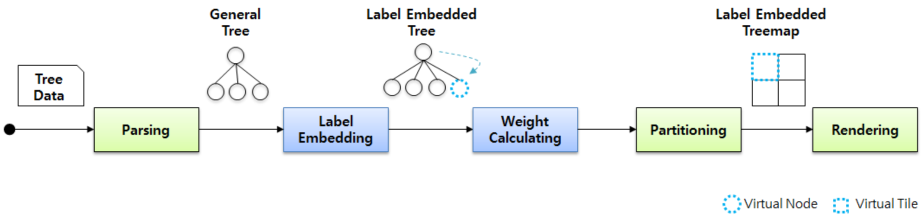

Fig. 3. Construction of a label embedded treemap

2.1 Tree Reconstruction

As parent nodes are placed into the treemap as leaf nodes without allocating specific spaces, a given parent node can neighbor its own child leaf nodes (Fig. 4). Each parent label node can be assigned to a treemap space without the use of an additional space allocation algorithm.

Because the parent nodes are inserted starting from the bottom of the tree structure, the label embedded tree is newly reconstructed from the general tree after each parent node insertion; however, this reconstruction causes no partitioning or rendering delay.

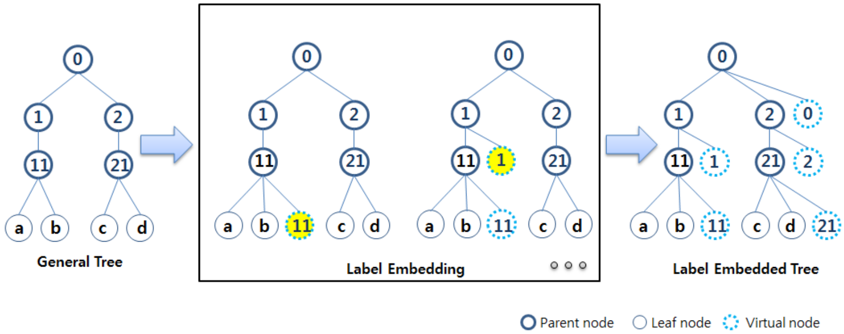

Fig. 4. Parent node insertion process

2.2 Calculating Weight Value of Parent's Label Node

As a newly inserted parent label node has no weight value, one must be generated following the label embedding process.

For a specific parent node has s children, let W_c represent the weight value of the k-th child, which can be formulated as the product of n, the number of leaf nodes contained by W_c, and α, a constant relating to the weight valence (formula (2)). The weight value for the parent node, W_p, is then the sum of all children nodes, which is calculated recursively using formula (1):

$$Wp = \sum_{c=1}^{s} \mathrm{Wc} \tag{1}$$

$$W\mathrm{c} = \alpha * n \tag{2}$$

2.3 Partitioning Using Label Embedded Tree

After the label embedding, re-constructed tree structure is just used for partitioning process. As a result newly added label node is also get a separate treemap space. (Fig 5)

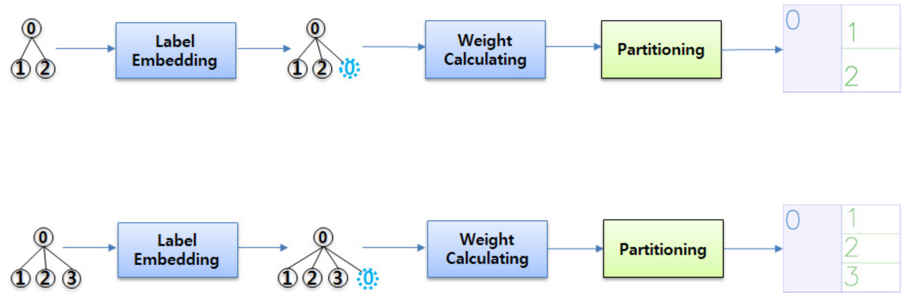

Fig. 5. Parent node also has a treemap space as tile

3 User Interaction

Faster node navigation is possible because of newly inserted parent node.

3.1 Highlighting Around Parent Node

To support the direct selection of parent node areas, focused nodes can be highlighted (Fig. 6). If a selected node is a label node corresponding to a parent, a focus line is formed around the area including the parent node, which helps the user to better understand the area being navigated.

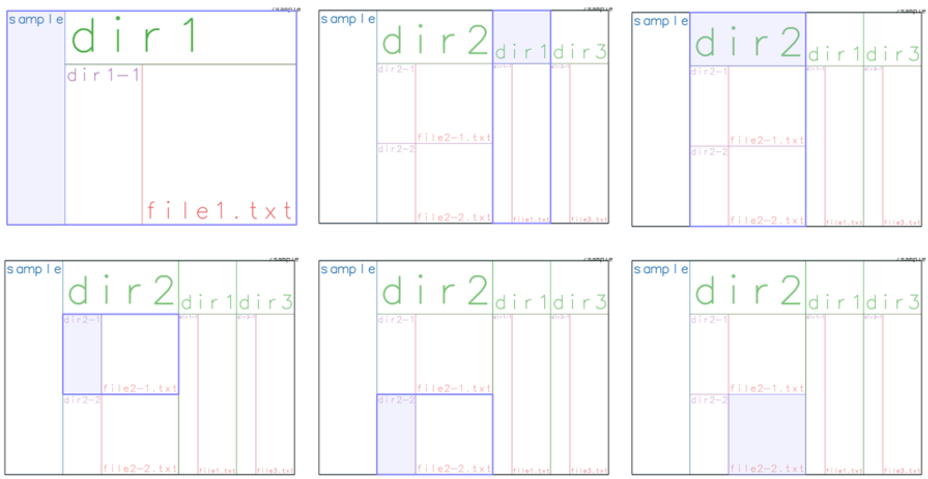

Fig. 6. Focused area is sized according to the focused node's children tree

3.2 Direct Node Selection and Smooth Animation Transition

After a node is selected, the selected area is expanded to full screen, as shown in Fig. 7. The zooming operation used to implement this process is completely consistent, causes no layout changes, and supports move rover animated transitions; essentially, this operation supports the same level of user interactivity as ZTM.

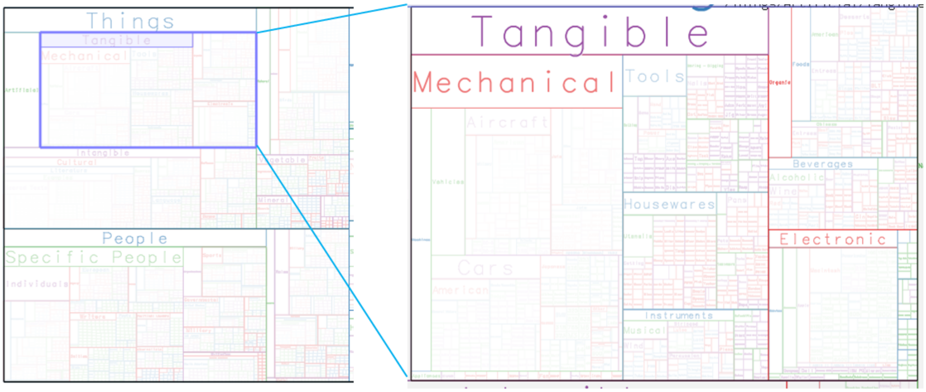

Fig. 7. Zoomed view corresponding to highlighted area

4 Evaluation

To confirm that our technique works correctly, we developed and tested an implementation of the proposed label embedded treemapping method using ZTM. We determined that we could generate label embedded trees that, following partitioning, could successfully function as label embedded treemaps for which rendered results could be generated. The transition animation process used for zooming was, of course, nearly identical to that used in ZTM, as it employed the same zooming operation based on the treemap structure. Finally, we confirmed that the focused node highlighting process functioned smoothly, making the process of direct node selection using highlighted focused label nodes quite useful for smooth treemap navigation.

A simple dataset containing three hierarchical nodes was rendered to produce a single leaf node, as shown in Fig. 9. It is seen that, although overlapped text labels occur at the top and bottom of the ZTM treemap space, there is no overlapping text between nodes on the label embedded treemap.

The rendering results generated by ZTM for a larger data set (Animalia [19], which has around 155,000 data items) are shown in Fig. 9, in which it is seen that many labels overlap. The rendering results produced by applying the label embedded treemap technique are shown in Fig. 11, in which no overlapping labels can be seen.

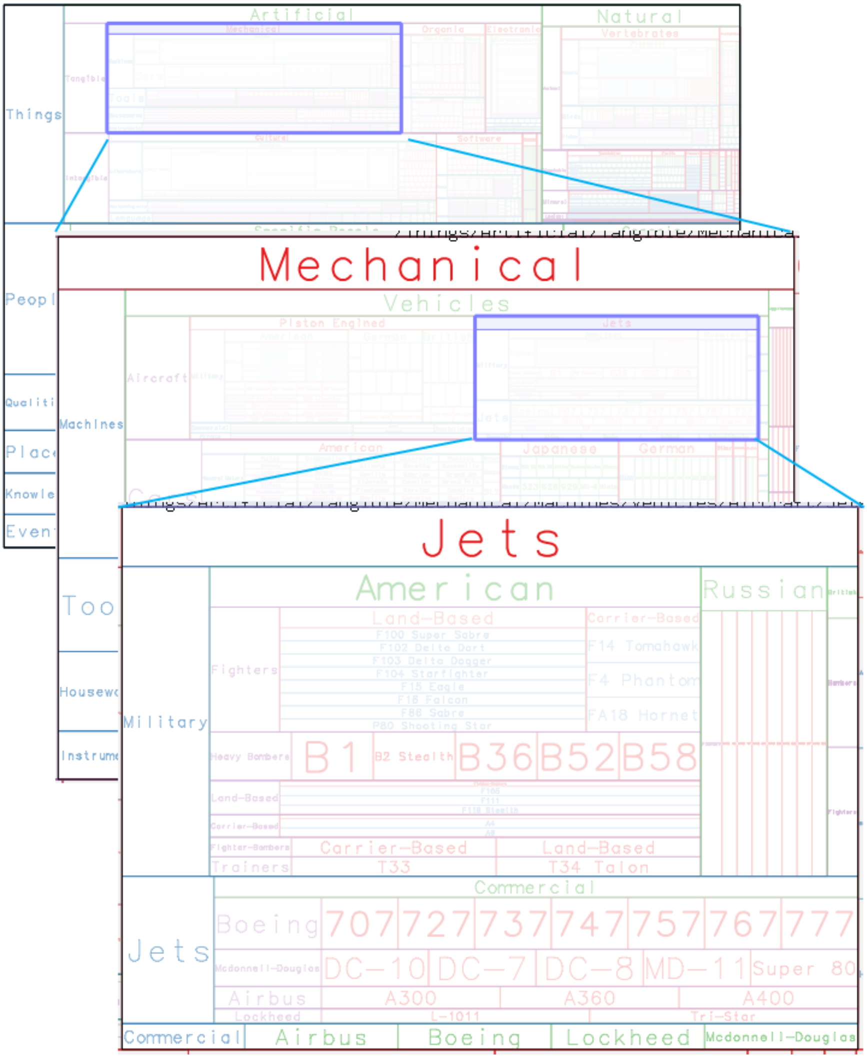

Fig. 8. Working example of direct node selection

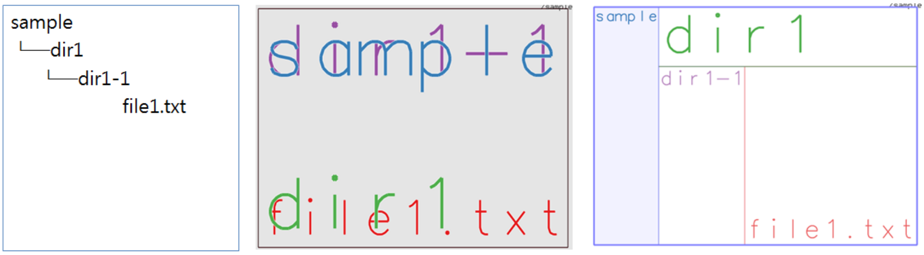

Fig. 9. (left) Simple dataset consisting of one file with three layers of folders. (middle) Rendering results using ZTM. (right) Rendering results using label embedded treemap.

Fig. 10. Slice-and-dice layout using ZTM with the “Animalia” dataset

Fig. 11. Slice-and-dice layout using label embedded ZTM with the "Animalia" dataset

5 Conclusion and Future Work

In this paper, we presented a label embedded treemap that avoids label overlapping and a user interaction technique based on the use of focused parent label nodes. The proposed label embedded treemapping technique avoids labeling conflicts between nodes and works very well with ZTM by exploiting advantages such as zoomable user interfaces and natural transition animation during zooming. Because there are no label overlaps between parent and leaf nodes, the proposed technique clearly navigates the treemap space, and because our method can support direct node selection using a focusable parent label node, navigation is easier than with ZTM.

As we focused primarily on resolving the text overlapping issues of ZTM and generating new user interaction potential based on the use of focusable parent label nodes, we were not able to conduct user testing with as much rigor as we would have liked, but our basic assessment showed positive results by performing data searching faster than ZTM. In order to improve data navigation, we plan to increase the label size and improve the readability of the parent label nodes, with the eventual goal of performing a controlled comparative evaluation between our label embedded treemapping technique and ZTM.

Acknowledgements. This research is supported by the Ministry of Culture, Sports and Tourism (MCST) and the Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program 2013.

References

1. Reingold, E.M., Tilford, J.S.: Tidier Drawings of Trees. IEEE Trans. Software Engineering SE-7(2), 223–228 (1981)
2. Johnson, B., Shneiderman, B.: Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures. In: Proc. IEEE Conference on Visualization 1991, pp. 284–291 (1991)
3. McGuffin, M.J., Robert, J.M.: Quantifying the Space-Efficiency of 2D Graphical Representations of Trees. Information Visualization 9(2), 115–140 (2010)
4. Baudel, T., Broeskema, B.: Capturing the Design Space of Sequential Space-Filling Layouts. IEEE Trans. Visualization and Computer Graphics 18(12), 2593–2602 (2012)
5. Fekete, J.D., Plaisant, C.: Interactive Information Visualization of a Million Items. In: Proc. IEEE Symposium on Information Visualization, pp. 117–124 (2002)
6. Bruls, M., Huizing, K., van Wijk, J.J.: Squarifiedtreemaps. In: Proceedings of Joint Eurographics and IEEE TCVG Symposium on Visualization (TCVG 2000), pp. 33–42 (2000)
7. Shneiderman, B., Wattenberg, M.: Ordered Treemap Layouts. In: Proceedings of the IEEE Symposium on Information Visualization (INFOVIS 2001), p. 73 (2001)
8. Bederson, B.B., Shneiderman, B., Wattenberg, M.: Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. ACM Transactions on Graphics (TOG) 21(4), 833–854 (2002)
9. van Wijk, J.J., van de Wetering, H.: Cushion treemaps: Visualization of hierarchical information. In: Proceedings of the IEEE Symposium on Information Visualization (INFOVIS 1999), pp. 73–78 (1999)
10. Elmqvist, N., Dragicevic, P., Fekete, J.: Color Lens: Adaptive Color Scale Optimization for Visual Exploration. IEEE Transactions on Visualization and Computer Graphics 17(6), 795–807 (2010)
11. Shi, K., Irani, P., Li, B.: An evaluation of content browsing techniques for hierarchical space-filling visualizations. In: Proceedings of IEEE Information Visualization (INFOVIS 2005), pp. 81–88 (2005)
12. Tu, Y., Shen, H.: Balloon Focus: a Seamless Multi-Focus+Context Method for Treemaps. IEEE Transactions on Visualization and Computer Graphics 14(6), 1157–1164 (2008)
13. Keahey, T.A.: Getting along: Composition of visualization paradigms. In: Proceedings of IEEE Information Visualization, INFOVIS 2001 (2001)
14. Perlin, K., Fox, D.: Pad: An Alternative Approach to the Computer Interface. In: Proc. ACM SIGGRAPH 1993, pp. 57–64 (1993)
15. Bederson, B.B.: PhotoMesa: A Zoomable Image Browser Using Quantum Treemaps and Bubblemaps. In: Proc. ACM Symposium on User Interface Software and Technology 2001, pp. 71–80 (2001)
16. Plaisant, C., Shneiderman, B.: Treemap 4.1.1 (2004), `http://www.cs.umd.edu/hcil/treemap/`
17. Blanch, R., Lecolinet, E.: Browsing ZoomableTreemaps: Structure-Aware Multi-Scale Navigation Techniques. IEEE Trans. Visualization and Computer Graphics 13(6), 1248–1253 (2007)
18. Kim, K.H., Han, J.H.: Label-embedded treemap. In: Proc. Korea Computer Congress 2013, pp. 1064–1066 (2013)
19. Blanch, R.: ZTM 1.0.2 (2012), `http://iihm.imag.fr/blanch/projects/ztm/src/`